MANUAL DE MANTENIMIENTO DEL SISTEMA

Diseño, Desarrollo e Implementación de una Plataforma Web para la Gestión de Servicios de Laboratorio

Versión: 1.0
Fecha: Noviembre 2025
Estado: Aprobado

Elaborado por	Revisado por	Fecha
Alexandra Tinjacá Cortés Departamento: Desarrollo	Alexandra Tinjacá Cortés y Juan Pablo Camacho Departamento: Calidad	Noviembre 2025 Departamento: TI

Registro de Control de Cambios

Versión	Fecha	Descripción del Cambio	Autor	Aprobado
1.0	Noviembre 2025	Versión inicial del manual	Alexandra Tinjacá Cortés	Todo el Equipo de Desarrollo LabPiloto

Cuadro 1: Historial de versiones del documento

Índice

Índice de figuras

Índice de cuadros

1 Introducción y Visión General

1.1 Propósito del Documento

Este manual establece los procedimientos, políticas y mejores prácticas para el mantenimiento del sistema Diseño, Desarrollo e Implementación de una Plataforma Web para la Gestión de Servicios de Laboratorio. Está dirigido a desarrolladores, administradores de sistemas y personal de soporte técnico.

1.2 Alcance del Sistema

El sistema comprende una plataforma web integral dockerizada que consta de tres componentes principales:

- **Frontend**: Plataforma web cliente desarrollada exclusivamente con tecnologías web estándar: HTML5, CSS3 y JavaScript . No se utilizan frameworks o librerías adicionales de frontend.
- **Backend**: Servicios API REST y lógica de negocio desarrollados con Spring Boot 3.x sobre Java 21, utilizando Maven (mvn) como gestor de dependencias y herramienta de construcción.
- **Base de Datos**: Sistema de gestión de base de datos PostgreSQL 15+ desplegado en Amazon RDS (Relational Database Service).

1.3 Características Técnicas Clave

Característica	**Implementación**
Arquitectura	Monolito dockerizado (Frontend + Backend)
Contenedorización	Docker (misma instancia)
Framework Backend	Spring Boot 3.x
Versión Java	Java 21
Gestor de Dependencias	Maven (mvn)
Frontend	HTML5 + CSS3 + JavaScript
Base de Datos	PostgreSQL 15+ en AWS RDS
Servidor Web Embebido	Tomcat (embebido en Spring Boot)

Cuadro 2: Stack tecnológico de la plataforma

1.4 Descripción de la Infraestructura

La plataforma opera bajo la siguiente configuración de infraestructura:

- **Instancia Única**: Frontend y Backend coexisten en la misma instancia dockerizada
- **Separación de Capas**: Base de datos externa en AWS RDS para mejor escalabilidad
- **Servicio Web**: Spring Boot actúa como servidor de aplicaciones y sirve los archivos estáticos del frontend
- **Comunicación**: Frontend se comunica con Backend mediante API REST sobre HTTP/HTTPS

1.5 Definiciones y Acrónimos

Término	**Definición**
CI/CD	Integración Continua/Despliegue Continuo
SLA	Acuerdo de Nivel de Servicio
API	Interfaz de Programación de Aplicaciones
SGBD	Sistema Gestor de Base de Datos
HTTP	Protocolo de Transferencia de Hipertexto
SSL	Capa de Conexiones Seguras

Cuadro 3: Glosario de términos técnicos

2 Arquitectura del Sistema

2.1 Diagrama de Arquitectura

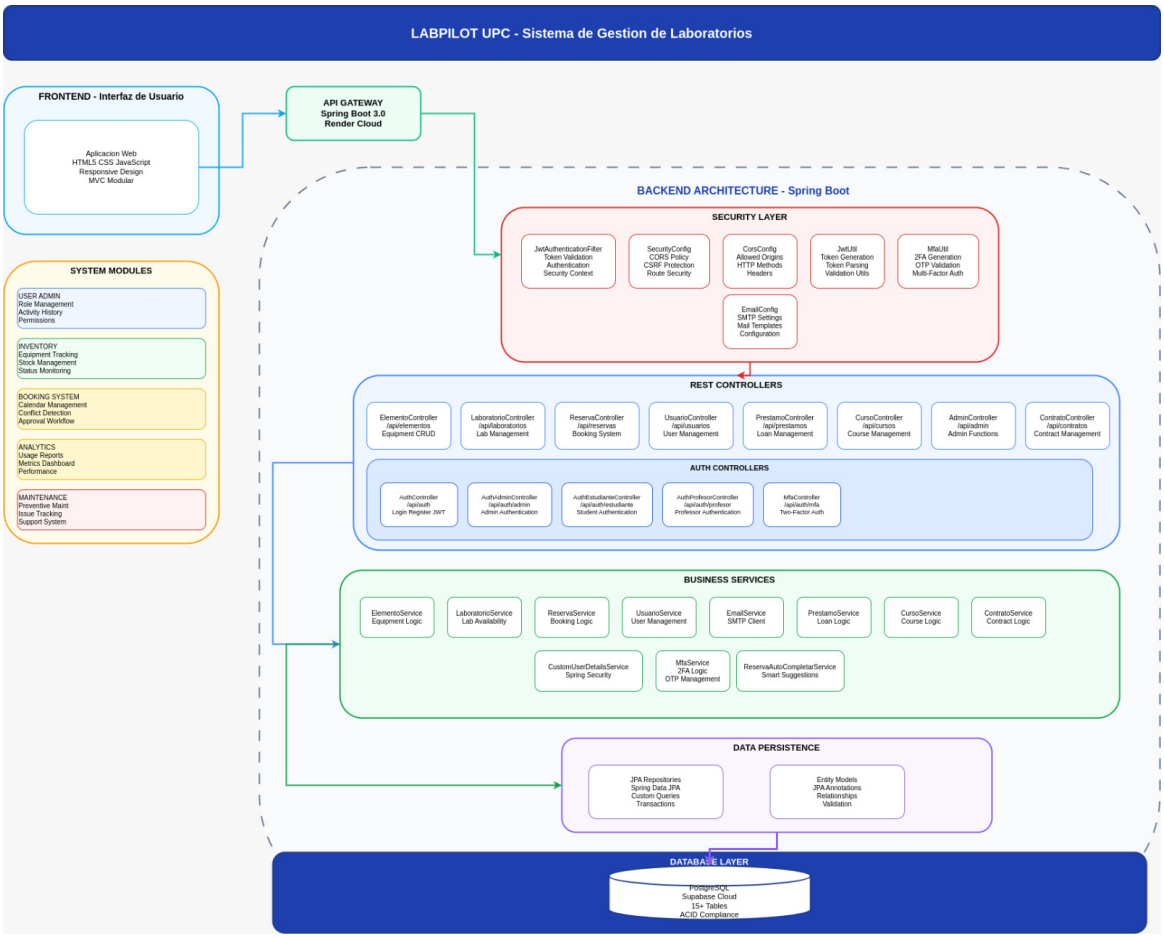

Figura 1: Diagrama de arquitectura completa del sistema

2.2 Componentes del Sistema

Componente	Tecnología	Versión	Responsable
Frontend Web	HTML5 + CSS3 + JavaScript	-	Equipo FullStack
Backend API	Spring Boot (Java)	3.x	Equipo Backend
Lenguaje Backend	Java	21	Equipo Backend
Base de Datos	PostgreSQL (AWS RDS)	15+	DBA/DevOps
Servidor Aplicaciones	Tomcat (embebido)	10.x	Equipo Backend
Contenedorización	Docker	-	DevOps
Gestor de Dependencias	Maven	3.8+	Equipo Backend

Cuadro 4: Matriz de tecnologías y responsables

2.3 Descripción de Componentes

- **Frontend Web**: Plataforma web construida con tecnologías nativas (HTML5, CSS3, JavaScript) sin frameworks adicionales
- **Backend API**: Servicios REST desarrollados con Spring Boot sobre Java 21, utilizando el patrón MVC
- **Base de Datos**: PostgreSQL gestionado mediante AWS RDS para alta disponibilidad y backups automáticos

- **Contenedorización**: Docker para empaquetado y despliegue consistente de la aplicación completa
- **Servidor Web**: Tomcat embebido en Spring Boot que sirve tanto los endpoints API como los archivos estáticos del frontend

3 Entornos y Configuración

3.1 Descripción de Entornos

Entorno	URL/Configuración	Propósito	Responsable
Desarrollo Local	`localhost:8080`	Desarrollo y pruebas locales	Desarrolladores
Entorno Docker Dev	Contenedor local en puerto 8080	Validar comportamiento en contenedor	Desarrolladores
Testing/QA	`localhost:8080,https://labpiloto.com`	Pruebas de calidad e integración	Equipo QA/Desarrollo
Producción	`https://labpiloto.com`	Entorno productivo para usuarios finales	DevOps

Cuadro 5: Entornos del sistema

3.2 Configuración por Entorno

- **Desarrollo Local**:
  - Spring Boot ejecutando directamente en IDE
  - Base de datos: PostgreSQL local o Docker
  - Configuración: `application-aws.properties`
- **Docker Development**:
  - Aplicación ejecutando en contenedor Docker
  - Base de datos: Contenedor PostgreSQL o RDS desarrollo
- **Producción**:
  - Contenedor en servidor/productivo
  - Base de datos: AWS RDS producción
  - Configuración: `application-aws.properties`

3.3 Configuración de Red y Seguridad

Servicio	Puerto	Protocolo	Acceso	Security Group
Aplicación Web (HTTP)	80	HTTP	Público (0.0.0.0/0)	sgr-082b867c509ded7d5
Aplicación Web (HTTPS)	443	HTTPS	Público (0.0.0.0/0)	sgr-08d1092563612df60
Spring Boot API	8080	HTTP	Interno/Privado	sgr-0b8323581e9eb2ded
SSH Administrativo	22	SSH	IP Específica (201.245.243.127/32)	sgr-07015695bdfec92a8
PostgreSQL RDS	5432	TCP	Solo Spring Boot	Configuración RDS

Cuadro 6: Configuración de puertos y reglas de seguridad AWS

3.4 Descripción de las Reglas de Seguridad

- **sgr-082b867c509ded7d5**: Permite tráfico HTTP público desde cualquier origen (0.0.0.0/0)
- **sgr-08d1092563612df60**: Permite tráfico HTTPS público desde cualquier origen (0.0.0.0/0)

- **sgr-07015695bdfec92a8**: SSH restringido a IP específica (201.245.243.127/32) para administración segura
- **sgr-0e01eb661b8ea3903**: SSH adicional configurado (revisar si es necesario)
- **sgr-0b8323581e9eb2ded**: Regla personalizada TCP - utilizado para el puerto 8080 de Spring Boot

3.5 Flujo de Tráfico

1. Los usuarios acceden vía HTTP/HTTPS (puertos 80/443) desde internet
2. El servidor web (Spring Boot embebido) escucha en puerto 8080 internamente
3. La aplicación Spring Boot se conecta a PostgreSQL RDS en puerto 5432
4. Administradores acceden vía SSH solo desde IP autorizada

3.6 Recomendaciones de Seguridad

- **Puerto 8080**: Debería estar restringido al balanceador de carga o instancia web
- **RDS PostgreSQL**: Configurar security group que solo permita conexiones desde la instancia de la aplicación
- **SSH**: Mantener restricción por IP específica para acceso administrativo
- Considerar implementar AWS WAF para protección adicional en puertos 80/443

4 Mantenimiento del Frontend

4.1 Requisitos del Sistema

```
Navegador Web Moderno (Chrome 90+, Firefox 88+, Safari 14+)
Servidor HTTP (Spring Boot embebido o similar)
```

Listing 1: Requisitos del sistema para desarrollo Frontend

4.2 Estructura del Proyecto

```
frontend/
          css/                # Hojas de estilo CSS
          js/                 # Logica JavaScript
             modules          /            # M dulos JavaScript modulares
                     AdministrarCursos.js
                     auth.js
                     gestor.js
                     inventario.js
                     laboratorios.js
                     prestamos.js
                     reportes.js
                     reservas.js
                     ui.js
                  admin.js
                  config.js
                  estudiantes.js
                  events.js
                  login.js
                  registrar-estudiante.js
                  registrar-admin.js
          *.html              # Paginas HTML principales
              admin.html
              index.html
```

```
24                  login-estudiante.html
25                  registrar-admin.html
26                  registrar-estudiante.html
```

Listing 2: Estructura de directorios del Frontend

4.3 Comandos Esenciales

Comando	Propósito	Entorno
`No requiere build`	Archivos estáticos listos	Todos
`Servir via Spring Boot`	Despliegue integrado	Producción
`http-server o similar`	Servidor desarrollo local	Desarrollo
`Validación HTML/CSS/JS`	Verificar estándares web	Desarrollo

Cuadro 7: Comandos de mantenimiento del Frontend

4.4 Descripción de Módulos JavaScript

- **auth.js**: Manejo de autenticación y autorización
- **admin.js**: Funcionalidades específicas de administrador
- **gestor.js**: Gestión general del sistema
- **inventario.js**: Control de inventario y equipos
- **laboratorios.js**: Gestión de laboratorios
- **prestamos.js**: Sistema de préstamos de equipos
- **reservas.js**: Sistema de reservas de laboratorios
- **reportes.js**: Generación de reportes y estadísticas
- **estudiantes.js**: Gestión de usuarios estudiantes
- **events.js**: Manejo de eventos del sistema
- **ui.js**: Componentes de interfaz de usuario
- **config.js**: Configuración global de la aplicación
- **registrar-estudiante.js**: Registrar usuario normal
- **registrar-admin.js**: Registrar usuario con rol de Administrador
- **login.js**: Gestión inicio de sesión
- **AdministrarCursos.js**: Administración cursos

4.5 Páginas HTML Principales

- **index.html**: Página principal/dashboard del sistema
- **login.html**: Página de autenticación de usuarios
- **admin.html**: Panel de administración
- **registrar-admin.html**: Registro de administradores

4.6 Características Técnicas

- **Arquitectura**: Aplicación web estática (HTML + CSS + JS)
- **Comunicación**: Consumo de APIs REST del backend Spring Boot
- **Almacenamiento**: Uso de localStorage/sessionStorage para datos temporales
- **Despliegue**: Servido como archivos estáticos por Spring Boot
- **Dependencias**: Sin frameworks externos (JavaScript puro)

5 Mantenimiento del Backend

5.1 Requisitos del Sistema

```
Java >= 21
Maven >= 3.8
Spring Boot >= 3.2
Docker >= 20.0
PostgreSQL >= 15
```

Listing 3: Requisitos del sistema para desarrollo Backend

5.2 Estructura del Proyecto

```
backend/labpilot/
          .mvn/wrapper/              # Maven Wrapper
          src/
                main/
                      java/edu/upc/labpilot/
                            LabPilotUpcApplication.java  # Clase principal
                            config/          # Configuraciones
                            controller/      # Controladores REST
                            dto/             # Objetos de Transferencia
                            model/           # Entidades JPA
                            repository/      # Repositorios Spring Data
                            service/         # L gica de negocio
                      resources/          # Recursos, propiedades
                test/java/edu/upc/labpilot/  # Pruebas
          mvnw                         # Maven Wrapper Linux
          mvnw.cmd                     # Maven Wrapper Windows
          pom.xml                      # Configuraci n Maven
```

Listing 4: Estructura de directorios del Backend Spring Boot

5.3 Comandos Esenciales

Comando	Propósito	Entorno
`docker build -t labpilot-backend:latest -f Dockerfile.backend .`	Contruir Backend	Todos
`docker run labpilot-backend:latest`	Correr Backend	Desarrollo
`docker build -t labpilot .`	Construir imagen Docker	Docker

Cuadro 8: Comandos de mantenimiento del Backend

5.4 Descripción de Paquetes

- **config**: Configuraciones de Spring (Security, CORS, Beans)
- **controller**: Endpoints REST API con anotaciones `@RestController`

- **dto**: Objetos Data Transfer Object para transferencia de datos
- **model**: Entidades JPA con anotaciones `@Entity` para mapeo BD
- **repository**: Interfaces Spring Data JPA `@Repository`
- **service**: Clases de servicio con lógica de negocio `@Service`

5.5 Archivos de Configuración Clave

- **pom.xml**: Dependencias Maven y configuración de build
- **application-aws.properties**: Configuración principal Spring Boot

5.6 Endpoints Críticos del API

Endpoint	Método	Propósito	SLA
`/api/auth`	POST	Autenticación de usuarios	99.9 %
`/api/usuarios`	GET, POST, PUT, DELETE	Gestión de usuarios	99.5 %
`/api/laboratorios`	GET, POST, PUT, DELETE	Gestión de laboratorios	99.8 %
`/api/elementos`	GET, POST, PUT, DELETE	Gestión de elementos/inventario	99.8 %
`/api/reservas`	GET, POST, PUT, DELETE	Sistema de reservas	99.9 %
`/api/prestamos`	GET, POST, PUT, DELETE	Sistema de préstamos	99.9 %
`/api/admins`	GET, POST, PUT, DELETE	Gestión de administradores	99.5 %

Cuadro 9: Endpoints críticos del API

5.7 Procedimiento de Despliegue

5.7.1 Parada y Limpieza de Contenedores Existentes

```
# Parar contenedores en ejecuci n
docker stop frontend backend

# Eliminar contenedores
docker rm frontend backend

# Limpiar im genes y redes no utilizadas
docker system prune -f
docker network prune -f
```

Listing 5: Parada y eliminación de contenedores existentes

5.7.2 Preparación de Infraestructura

```
# Crear red personalizada para la aplicaci n
docker network create labpilot-network
```

Listing 6: Creación de red Docker para comunicación entre contenedores

5.7.3 Despliegue del Backend

```
# Construir imagen del backend
docker build -t labpilot-backend:latest -f Dockerfile.backend .

# Ejecutar contenedor del backend
docker run -d \
  --name backend \
  --network labpilot-network \
```

```
  -p 8080:8080 \
  -e SPRING_PROFILES_ACTIVE=aws \
  labpilot-backend:latest
```

Listing 7: Construcción y despliegue del contenedor Backend

5.7.4 Despliegue del Frontend

```
# Construir imagen del frontend
docker build -t labpilot-frontend:latest -f Dockerfile.frontend.ssl .

# Ejecutar contenedor del frontend
docker run -d \
  --name frontend \
  --network labpilot-network \
  -p 80:80 \
  -p 443:443 \
  -v /etc/letsencrypt:/etc/letsencrypt \
  labpilot-frontend:latest
```

Listing 8: Construcción y despliegue del contenedor Frontend

5.7.5 Verificación del Despliegue

```
# Verificar logs del backend
docker logs backend

# Verificar logs del frontend
docker logs frontend

# Verificar contenedores en ejecuci n
docker ps

# Verificar salud de la aplicaci n
curl http://localhost:8080/api/health
```

Listing 9: Verificación del estado de los contenedores

5.8 Variables de Entorno Críticas

Variable	Contenedor	Propósito
`SPRING_PROFILES_ACTIVE=aws`	Backend	Perfil Spring para AWS
`/etc/letsencrypt`	Frontend	Certificados SSL/TLS
`labpilot-network`	Ambos	Red interna de comunicación

Cuadro 10: Variables de entorno y configuraciones críticas

5.9 Consideraciones de Seguridad

- Los certificados SSL se montan como volumen de solo lectura
- La red `labpilot-network` aísla la comunicación entre contenedores
- El perfil `aws` configura conexiones seguras a RDS
- Los puertos 80 y 443 están expuestos para acceso web
- El puerto 8080 del backend es accesible solo internamente

5.10 Script de Despliegue Automatizado

```
#!/bin/bash
set -e

echo "=== Iniciando despliegue de LabPilot ==="

# Parar y limpiar contenedores existentes
echo "Parando contenedores..."
docker stop frontend backend 2>/dev/null || true
docker rm frontend backend 2>/dev/null || true

# Limpiar sistema
echo "Limpiando recursos Docker..."
docker system prune -f
docker network prune -f

# Crear red
echo "Creando red labpilot-network..."
docker network create labpilot-network 2>/dev/null || true

# Desplegar backend
echo "Construyendo y desplegando backend..."
docker build -t labpilot-backend:latest -f Dockerfile.backend .
docker run -d --name backend --network labpilot-network -p 8080:8080 -e
    SPRING_PROFILES_ACTIVE=aws labpilot-backend:latest

# Desplegar frontend
echo "Construyendo y desplegando frontend..."
docker build -t labpilot-frontend:latest -f Dockerfile.frontend.ssl .
docker run -d --name frontend --network labpilot-network -p 80:80 -p 443:443 -v /etc/
    letsencrypt:/etc/letsencrypt labpilot-frontend:latest

echo "=== Despliegue completado ==="
echo "Verificando contenedores..."
docker ps
```

Listing 10: Script completo de despliegue automatizado

6 Mantenimiento de Base de Datos

6.1 Esquema de Base de Datos

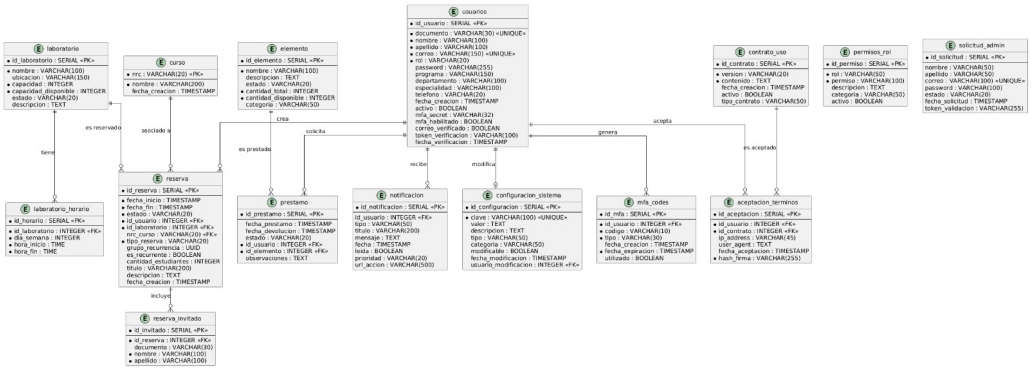

Figura 2: Diagrama Entidad-Relación de la base de datos

6.2 Procedimiento de Migración de Base de Datos

Paso	Descripción	Responsable	Tiempo
1	Backup RDS PostgreSQL	DevOps	15 min
2	Validar scripts Flyway/Liquibase	Desarrollo	10 min
3	Ejecutar migración Spring Boot	DevOps	5 min
4	Verificar integridad datos	Desarrollo	15 min
5	Rollback si es necesario	DevOps	10 min

Cuadro 11: Procedimiento de migración de base de datos

6.2.1 Migraciones con Spring Boot

```
# Las migraciones se ejecutan automaticamente al iniciar la aplicacion
# Archivos en: src/main/resources/db/migration/

# V1__Create_users_table.sql
# V2__Create_laboratories_table.sql
# V3__Add_indexes.sql

# Verificar migraciones aplicadas
SELECT * FROM flyway_schema_history;
```

Listing 11: Ejemplo de migración con Flyway integrado en Spring Boot

7 Monitoreo y Alertas

7.1 Métricas Clave

Métrica	Umbral Crítico	Umbral Advertencia	Acción
CPU Docker	>90 %	>80 %	Revisar contenedores
Memory Docker	>95 %	>85 %	Optimizar/Reiniciar
Disk RDS	<5 % libre	<10 % libre	Expandir almacenamiento
Response Time API	>3s	>1s	Optimizar consultas
Error Rate Spring Boot	>5 %	>2 %	Revisar logs
Conexiones DB	>90 % max	>70 % max	Ajustar pool

Cuadro 12: Métricas de monitoreo del sistema

7.2 Monitoreo de Contenedores Docker

Contenedor	Métricas	Herramienta
Backend Spring Boot	CPU, Memoria, Logs	docker stats, docker logs
Frontend Nginx	Requests, SSL, Latencia	Nginx status
PostgreSQL RDS	Connections, Locks, I/O	AWS CloudWatch

Cuadro 13: Monitoreo específico por contenedor

7.3 Alertas Recomendadas

- **Contenedor caído**: Cuando backend/frontend no responden
- **Base de datos**: Conexiones máximas alcanzadas
- **SSL**: Certificado próximo a expirar
- **Storage RDS**: Espacio en disco crítico
- **Error 5xx**: Tasa de errores HTTP elevada

7.4 Comandos de Monitoreo en Tiempo Real

```
# Monitorear recursos de contenedores
docker stats

# Ver logs en tiempo real
docker logs -f backend
docker logs -f frontend

# Verificar salud de la aplicaci n
curl -s http://localhost:8080/actuator/health | jq .

# Monitorear base de datos RDS
# Via AWS Console -> RDS -> Monitoring
```

Listing 12: Comandos para monitoreo manual del sistema

7.5 Checklist de Diagnóstico

1. Verificar estado de servidores y servicios
2. Revisar logs de aplicación y sistema
3. Comprobar conectividad de base de datos
4. Validar certificados SSL y DNS
5. Verificar consumo de recursos
6. Revisar métricas de monitoreo

8 Plan de Continuidad del Negocio

8.1 Procedimiento de Recuperación

Escenario	RTO	RPO	Procedimiento
Fallo aplicación	30 min	5 min	Restaurar desde backup
Fallo base datos	2 horas	15 min	Recuperar último backup
Fallo servidor	1 hora	10 min	Activar servidor standby
Desastre completo	4 horas	1 hora	Recuperación completa

Cuadro 14: Objetivos de recuperación

8.2 Backup y Restauración

```
-- Detener aplicaciones conectadas a la BD
-- Ejecutar restore
pg_restore -U postgres -h localhost -d sistema_prod \
--clean --if-exists backup_file.dump

-- Verificar integridad
SELECT COUNT(*) FROM information_schema.tables;
SELECT COUNT(*) FROM usuarios;
SELECT COUNT(*) FROM pedidos;
```

Listing 13: Procedimiento de restauración de base de datos

9 Anexos

9.1 Enlaces de Interés

- **Repositorio GitHub**: https://github.com/Epifis/LabPiloto.git

Referencias

1. IEEE Std 1063-2001 - Software User Documentation
2. IEEE Std 829-2008 - Software Test Documentation
3. ISO/IEC/IEEE 12207 - Systems and software engineering